A 140-Mb/s, 32-state, radix-4 Viterbi decoder

Black, P.J. Meng, T.H.
Inf. Syst. Lab., Stanford Univ., CA;

This paper appears in: **Solid-State Circuits, IEEE Journal of**
On page(s): 1877-1885
Volume: 27, Issue: 12, Dec 1992
ISSN: 0018-9200
References Cited: 19
CODEN: IJSCBC
INSPEC Accession Number: 4338291

Abstract:
A 140-Mb/s, 32-state, radix-4, R=1/2, eight-level soft-decision Viterbi decoder has been designed and fabricated using 1.2-μm double-metal CMOS. The architecture of the add-compare-select (ACS) array is based on a restructuring of the conventional radix-2 trellis into a radix-4 trellis. Radix-4 units, consisting of four 4-way ACS units, process two stages of the constituent radix-2 trellis per iteration. A four-way ACS circuit achieves an iteration delay 17% longer than comparable two-way ACS circuits, resulting in a factor of 1.7 increase in throughput. A ring-based ACS placement and state metric routing topology achieves an area efficiency comparable to radix-2 designs. In a process referred to as pretrace-back, one stage of lookahead is applied to the trace-back recursion, combining two radix-4 trace-back iterations into a single radix-16 iteration based on 4-b decisions. This allows implementation of trace-back using one compact, single-ported decision memory, organized as a cyclic buffer. A 7.30-mm×8.49-mm chip containing 146000 transistors achieves a radix-4 iteration rate of 70 MHz

Index Terms:
CMOS integrated circuits decoding integrated logic circuits shift registers trellis codes 1.2 micron 140 Mbit/s 32-state decoder add compare select array architecture binary shift register trellis cyclic buffer double-metal CMOS eight-level soft-decision Viterbi decoder iteration delay lookahead pretrace-back radix-16 iteration radix-4 Viterbi decoder radix-4 trace-back iterations radix-4 trellis single-ported decision memory state metric routing topology throughput trace-back recursion

A 140-Mb/s, 32-State, Radix-4 Viterbi Decoder

Peter J. Black, *Student Member, IEEE*, and Teresa H. Meng, *Member, IEEE*

Abstract—**A 140-Mb/s, 32-state, radix-4, $R = 1/2$, eight-level soft-decision Viterbi decoder has been designed and fabricated using 1.2-μm double-metal CMOS. The architecture of the add-compare-select (ACS) array is based on a restructuring of the conventional radix-2 trellis into a radix-4 trellis. Radix-4 units, consisting of four 4-way ACS units, process two stages of the constituent radix-2 trellis per iteration. A 4-way ACS circuit is used that achieves an iteration delay 17% longer than comparable two-way ACS circuits, resulting in a factor of 1.7 increase in throughput. A ring-based ACS placement and state metric routing topology is described for the radix-4 ACS array, which achieves an area efficiency comparable to radix-2 designs. In a process referred to as pretrace-back, one stage of lookahead is applied to the trace-back recursion, combining two radix-4 trace-back iterations into a single radix-16 iteration based on 4-b decisions. This allows implementation of trace-back using one compact, single-ported decision memory, organized as a cyclic buffer. The 7.30-mm × 8.49-mm chip containing 146 000 transistors achieves a radix-4 iteration rate of 70 MHz, equivalent to a decode rate of 140 Mb/s under typical operating conditions ($V_{DD} = 5.0$ V, $T_A = 27°C$).**

I. Introduction

In recent years there has been interest in implementations of the Viterbi algorithm at rates on the order of 100 Mb/s. Driving applications include convolutional decoders for error correction, trellis code demodulation for communication channels, and digital sequence detection for magnetic storage devices. An important problem found in such applications is the decoding of a binary shift register (BSR) trellis. The classical high throughput implementation for such decoders is the radix-2 fully parallel approach, where add-compare-select (ACS) units are assigned to each state and organized in pairs to iterate one stage of a two-state trellis. The decode rate of this approach is fundamentally limited by either the recursive ACS iteration or the recursive trace-back iteration. In commercial decoder designs, implementing the trace-back portion of the algorithm in an area-efficient manner often results in the trace-back recursion being the limiting critical path. To date, such single chip implementations have achieved decode rates around 25 Mb/s [1], [2].

The most common technique for increasing decoder throughput is to run multiple decoders in parallel on either interleaved data [3] or blocked data [4], [5]. The resulting architecture achieves a speedup that is directly proportional to the additional complexity and is referred to as ideal linear scaling. Using decoder level parallelism to meet a throughput specification results in an overall area efficiency approximately equal to that of the core decoder.

An alternative approach to high throughput design is to exploit new forms of concurrency within a decoder by reformulation of the algorithm [6]. However, given the option of decoder-level parallelism, such architectures should achieve an area efficiency comparable to existing designs. Otherwise, the increased throughput could have been matched in less area by simply running existing decoders in parallel. To date, the most area-efficient designs are based on the classical fully parallel radix-2 architecture. In this paper we show that the throughput of this architecture can be extended by applying one stage of lookahead to both the ACS and trace-back recursions. The result is an ACS iteration based on a radix-4 trellis and trace-back iteration based on a radix-16 trellis. This architecture is demonstrated in a 32-state, $R = 1/2$ Viterbi decoder that achieves a decode rate of 140 Mb/s.

A brief synopsis of this work has been presented in [7]. In Section II the Viterbi algorithm and higher radix formulations for a BSR trellis are described. Section III discusses the decoder specification and implementation details of all major functional blocks. In Section IV the ACS placement and the state metric interconnect routing strategy for a radix-4 trellis are addressed. Finally, in Section V fabrication results are presented.

II. Viterbi Algorithm

The Viterbi algorithm is an optimum algorithm for estimating the state sequence of a finite state process given a set of noisy observations. An important class of problem is the BSR process, the state of which can be described by the contents of a binary shift register. An $R = 1/2$ convolutional encoder is based on a binary shift register. For each input bit, two output bits are generated as a modulo 2 combination of the shift register contents and the input. Given a corrupted encoded sequence, the decoder uses the Viterbi algorithm to estimate the state sequence of the encoder, for which the original input sequence is decoded.

This section is an overview of the Viterbi algorithm for a BSR process of memory length v. We adopt the convention that the state S_n of the process is given by the shift register contents interpreted as a radix-2 number, with the MSB corresponding to the oldest sample. A complete discussion of the Viterbi algorithm can be found in [8].

Manuscript received April 26, 1992; revised July 29, 1992. This work was supported in part by an IBM Graduate Fellowship and DARPA.

The authors are with the Information Systems Laboratory, Stanford University, Stanford, CA 94305.

IEEE Log Number 9203616.

A. Decoding a Radix-2 Trellis

The Viterbi algorithm is typically expressed in terms of a trellis diagram, which is a time indexed version of the state diagram. The simplest trellis is shown in Fig. 1(a) for a two-state BSR process. At time $n-1$ there are two possible states, each of which can transition to either of the two next states at time n depending on the binary input. Each transition is weighted according to its likelihood based on the noisy observations of the process. The Viterbi algorithm determines the most likely state sequence by finding the minimum weight path, or shortest path, through the trellis.

Associated with each trellis state S at time n is a state metric Γ_n^S, which is the accumulated metric along the shortest path leading to that state, and a decision d_n^S, which identifies the entering transition on the shortest path. For a 2^v-state BSR process it is convenient to reorganize the radix-2 trellis from $n-1$ to n into 2-state radix-2 subtrellises of the form shown in Fig. 1(a). Each state transition is identified by the previous state ix and the next state xj, where i is the bit shifted out of the shift register, j is the next input bit, and x represents the common state bits. The likelihood of this transition is given by the branch metric λ^{ixj}. Given the state metrics of the two predecessor states at time $n-1$, the state metric for state $x0$ at time n is given by

$$\Gamma_n^{x0} = \min\left(\Gamma_{n-1}^{0x} + \lambda_{n-1}^{0x0}, \Gamma_{n-1}^{1x} + \lambda_{n-1}^{1x0}\right) \quad (1)$$

and the decision is given by the state bit (i) shifted from the predecessor state that yielded the minimum updated metric. This recursive update is the well-known ACS operation and is implemented using a 2-way ACS unit as shown in Fig. 1(b). Since the two output states of the subtrellis have common predecessor states, it is logical to combine two 2-way ACS units into a radix-2 unit, which updates that state metrics for the radix-2 subtrellis as shown in Fig. 1(c).

The input sequence is decoded using a recursive procedure known as trace-back. Given an arbitrary starting state S_n and its decision d_n^S, the previous state is given by $S_{n-1} = d_n^S(S_n >> 1)$, which corresponds to a 1-b right shift of the current shift register state with input equal to the current state decision. For example, given $S_n = 00111$ and $d_n^S = 1$, the previous state is $S_{n-1} = 10011$. This recursion proceeds to a depth L known as the survivor path length, at which point all the shortest paths (survivor paths) from all possible starting states should have merged, and the input corresponding to the transition from the state at time index $n-L$ is decoded.

B. Radix-4 Trellis ACS Update

Using the unified approach to state metric update [9], a 2^v-state trellis can be iterated from time index $n-k$ to n by decomposing the trellis into 2^{v-k} subtrellises, each consisting of k iterations of a 2^k-state trellis. Each 2^k-state subtrellis can be collapsed into an equivalent one-stage radix-2^k trellis by applying k levels of lookahead to the recursive ACS update [10], [11]. Collapsing the trellis does not affect decoder performance since there is a one-to-one mapping between the shortest path in the collapsed trellis and the original radix-2 trellis; hence the decoded paths are identical. An example of the decomposition for an eight-state radix-2 trellis into an equivalent radix-4 trellis using one stage of lookahead is shown in Fig. 2.

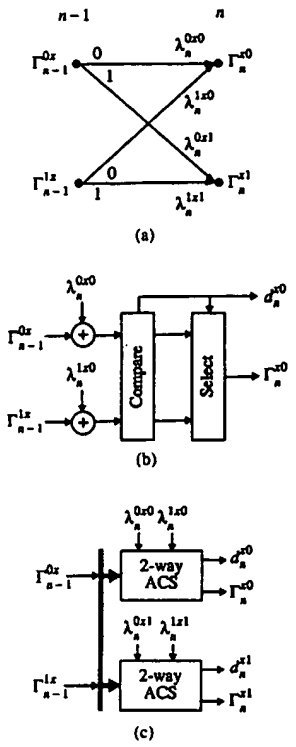

Fig. 1. (a) Radix-2 trellis. (b) 2-way ACS. (c) Radix-2 ACS unit.

The iteration time of a state-parallel Viterbi decoder is limited to the iteration time of the ACS. For a radix-2^k implementation, the effective iteration time is $1/k$ times the delay through a 2^k-way ACS, since k iterations of the original radix-2 trellis are processed per iteration. If the delay through a 2^k-way and 2-way ACS are equal, a potential k fold speedup is achievable for a complexity increase of 2^{k-1} in the ideal case.

A summary of the potential speedup and complexity as a function of trellis radix is given in Table I. In practice the ideal speedups cannot be achieved due to the additional delay overhead associated with the exponentially increasing number of inputs to the compare. The radix-4 architecture is of special interest because it corresponds to the intersection of the exponentially increasing complexity curve and the ideal linear speedup curve. As a result, the radix-4 architecture was chosen because it is the only architecture to offer a throughput increase while maintaining area efficiency.

The state metric labelling for a radix-4 trellis is shown in Fig. 3(a). Since each output state has four predecessor

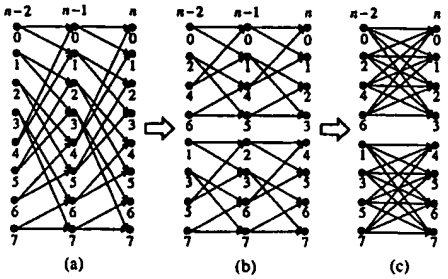

Fig. 2. (a) 8-state radix-2 trellis. (b) 4-state subtrellis decomposition. (c) 8-state radix-4 trellis.

TABLE I
RADIX-2^k COMPLEXITY SPEED MEASURES

Radix	k	Ideal Speedup	Complexity Increase	Area Efficiency
2	1	1	1	1
4	2	2	2	1
8	3	3	4	0.75
16	4	4	8	0.5

states, state metrics are updated using a four-way ACS unit as shown in Fig. 3(b) and decisions consist of 2 b. Each radix-4 subtrellis is updated using four 4-way ACS units, which are combined to form a radix-4 ACS unit as shown in Fig. 4.

C. Radix-16 Trellis Trace-Back Update

One stage of lookahead can be applied to the trace-back recursion to ease the timing constraints on this potential critical path. For a radix-4 trellis the trace-back iterations from n to $n-2$ and from $n-2$ to $n-4$ can be expressed as

$$S_{n-2} = d_n^S(S_n \gg 2)$$
$$S_{n-4} = d_{n-2}^S(S_{n-2} \gg 2). \quad (2)$$

Applying one stage of lookahead, these radix-4 trace-back iterations can be combined into a single radix-16 iteration from n to $n-4$ as follows:

$$\begin{aligned} S_{n-4} &= d_{n-2}^S(d_n^S(S_n \gg 2) \gg 2) \\ &= d_{n-2}^S d_n^S(S_n \gg 4) \\ &= d_{n,n-2}^S(S_n \gg 4) \end{aligned} \quad (3)$$

where $d_{n,n-2}^S$ is the composite 4-b radix-16 decision. For example, given the current state decision is $d_n^S = 01$ and the decision of the previous state as selected by this decision is $d_{n-2}^S = 11$, the composite decision is $d_{n,n-2}^S = 1101$. The composite decisions are calculated prior to running the trace-back algorithm and hence must be calculated for all states. We refer to the calculation of the composite decisions as *pretrace-back*.

Formation of the pretraced decision for a state requires both the current state decision and the decisions of all possible previous states. The current state decision defines

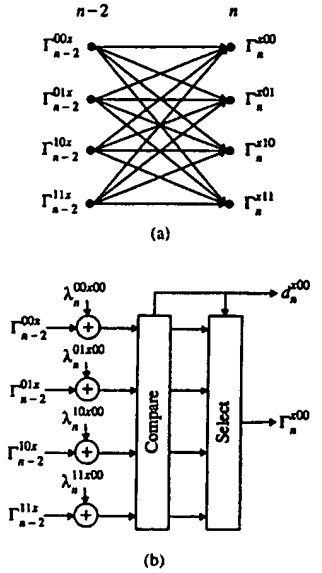

Fig. 3. (a) Radix-4 trellis. (b) 4-way ACS.

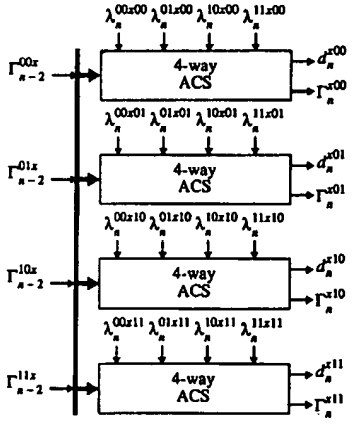

Fig. 4. Radix-4 ACS unit.

the previous state and hence can be used to select the previous state decision. The selected previous decision and the current decision are then combined to form the composite 4-b radix-16 decision. This pretrace-back operation is implemented simply using a multiplexer as shown in Fig. 5(a). We refer to this logical unit as four-way pretrace-back unit. Since the four output states of a radix-4 trellis have the same four previous states, a radix-4 pretrace-back unit consisting of four 4-way pretrace-back units logically maps to a radix-4 subtrellis as shown in Fig. 5(b).

The size of the decision memory is unchanged by the use of pretrace-back since 4-b decisions are stored for each state for every other ACS iteration, as opposed to 2-b decisions per state per iteration. Using radix-16 based trace-

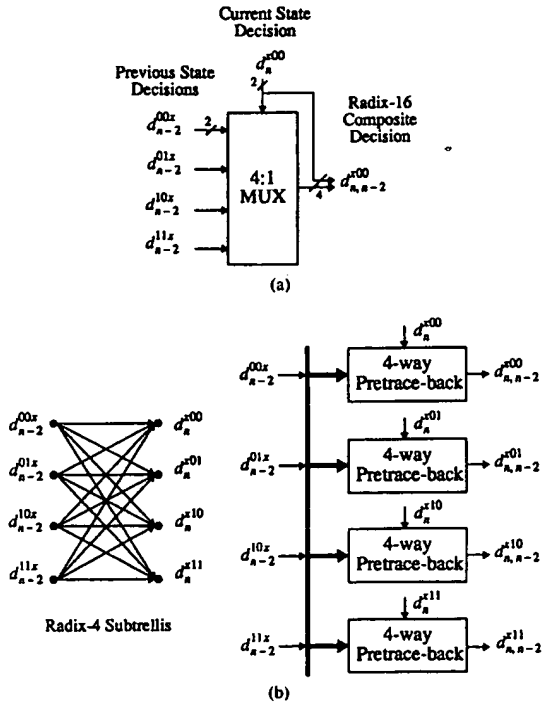

Fig. 5. (a) 4-way pretrace-back unit. (b) Radix-4 pretrace-back unit.

back, 4 b are decoded per iteration at one quarter the rate of conventional radix-2 implementations.

D. Modulo Arithmetic for ACS

The recursive state metric update results in unbounded word growth due to the addition of branch metrics, which are always nonnegative. We avoid normalization using the modulo arithmetic approach proposed in [12]. Modulo arithmetic exploits the fact that the Viterbi algorithm inherently bounds the maximum dynamic range Δ_{max} of state metrics to be

$$\Delta_{max} \leq \lambda_{max} \log_2 N \tag{4}$$

where N is the number of states and λ_{max} is maximum branch metric for the original radix-2 trellis [13]. Given two numbers a and b such that $|a - b| < \Delta$, which are to be compared using subtraction, a result from number theory states that the comparison can be evaluated as $(a - b) \bmod 2\Delta$ without ambiguity. Hence, the state metrics can be updated and compared modulo $2\Delta_{max}$. By appropriately choosing the state metric precision, the modulo arithmetic is implicitly implemented by ignoring the state metric overflow.

The required state metric precision is equal to twice the maximum dynamic range of the updated state metrics just before the compare stage and the number of bits required is given by

$$\Gamma_{bits} = \lceil \log_2 (\Delta_{max} + k\lambda_{max}) \rceil + 1. \tag{5}$$

The term $k\lambda_{max}$ accounts for the potential dynamic range increase from the input of the radix-2^k ACS to the input of the compare stage due to the branch metric addition. The equivalent of overflow errors will result if the design value for the dynamic range is exceeded during operation; hence, the upper bound given in (4) is used to ensure correct operation under all SNR conditions. For the 32-state radix-4 decoder implemented, $k = 2$, $\lambda_{max} = 14$, $\Delta_{max} = 70$, and the required state metric precision is 8 b.

III. Decoder Implementation

A. Specification

The specifications for the Viterbi decoder implemented are:

- $K = 6$ (32 states), $R = 1/2$ convolutional decoder,
- Generator polynomials $G_1 = 111011$ and $G_2 = 110001$ (coding gain of 4.8 dB at 10^{-5} BER),
- 8-level soft decision inputs,
- survivor path length 32 (slightly greater than 5 times the constraint length [14]).

A block diagram of the complete decoder is shown in Fig. 6. In the following subsections, implementation details of all major blocks are described.

B. Branch Metric Unit

Branch metrics for the radix-4 trellis are generated by combining branch metrics of successive iterations of the underlying radix-2 trellis. For each iteration of the radix-2 trellis, two 3-b soft decision inputs G_2G_1 are combined to form four possible branch metrics $\lambda_n(G_2G_1)$ corresponding to the four possible encoder outputs $(G_2G_1) \in \{(00), (01), (10), (11)\}$. Branch metrics are generated using a uniform distance measure equal to the symbol itself when compared to a logic ZERO and its one's complement when compared to a logic ONE [14]. Although not implemented, the radix-4 architecture can support punctured codes by simply forming the radix-2 metrics as in a normal radix-2 punctured decoder. A block diagram of the radix-2 branch metric unit is shown in Fig. 7(a).

The 16 possible radix-4 branch metrics for iteration $n - 2$ to n are formed from the Cartesian product set $\lambda_{n-1}(G_2G_1) \times \lambda_n(G_2G_1)$ of the radix-2 metrics for iterations $n - 1$ and n. In this design the radix-4 metrics are centrally calculated, then globally distributed to the ACS units. However, since the number of radix-4 metrics is the square of the number of radix-2 metrics, in some cases (e.g., $R = 1/4$ code) it is more area efficient to globally distribute the radix-2 metrics and locally calculate the radix-4 metrics as required in the ACS units. The block diagram of the radix-4 branch metric unit is shown in Fig. 7(b).

The complete branch metric unit (BMU) consists of two radix-2 metric units operating in parallel for iterations $n - 1$ and n, the outputs of which are combined in the radix-4 metric unit. Appropriate pipeline stages have been added to ensure the metric generation is not the limiting

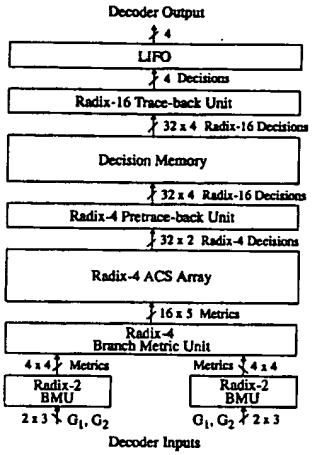

Fig. 6. Decoder block diagram.

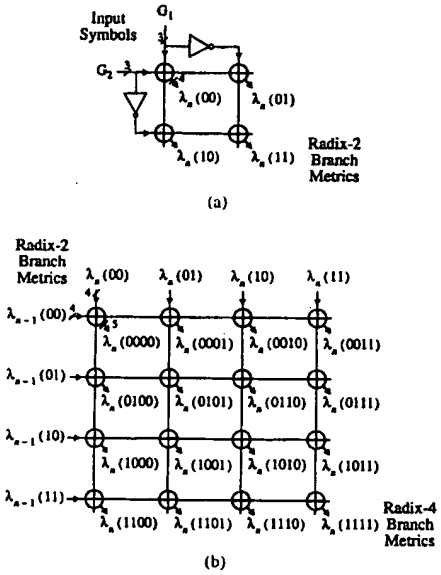

Fig. 7. (a) Radix-2 branch metric unit. (b) Radix-4 branch metric unit.

critical path. Given 3-b soft decision inputs, 4 b of precision are required for the radix-2 metrics and 5 b for the radix-4 metrics.

C. Radix-4 ACS Unit

In order to achieve the potential twofold increase in throughput offered by the radix-4 architecture, the 4-way ACS delay must be equal to that of a 2-way ACS. Fast, area-efficient 2-way ACS units are designed using ripple carry arithmetic for the add and compare operations. Faster adder structures offer little speed advantage at the required precision of 8 b and do not warrant the additional area overhead, especially when the combined delay of the add and compare are considered. By implementing the compare using subtraction, the adder and subtractor carry chains run in parallel from LSB to MSB, resulting in an add-compare delay that is only one full adder bit delay longer than the 8-b ripple carry add delay alone.

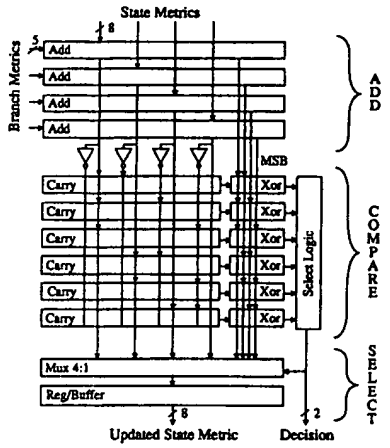

Fig. 8. 4-way ACS block diagram.

The block diagram of the 4-way ACS used to achieve comparable delay to the reference 2-way ACS is shown in Fig. 8. The 4-way compare is evaluated by generating the six possible pairwise comparisons and combining the results in two levels of logic to form the minimum metric selection. Only the subtractor carry chains were implemented to generate the sign of results and hence the comparison. The adder carry chain circuit is based on the conventional six-transistor generate-propagate (GP) style static logic gate shown in Fig. 9(a). For the subtractors, the overhead of GP generation logic was eliminated using the equivalent ten transistor compound gate shown in Fig. 9(b). This structure is slower than the adder circuit; however, since the subtractor carry chains are unloaded, equalization of the two carry chain critical paths was achieved.

Compared to the benchmark 2-way ACS, the delay through the 4-way ACS is increased due to three factors: the increased fan-out of the adder outputs, the additional logic in generating the multiplexer control signals from pairwise comparisons, and the additional delay of 4:1 multiplexer compared to a 2:1 multiplexer. Overall the 4-way ACS delay is 17% longer than the 2-way ACS of a similar structure, resulting in a speedup factor of 1.7 in our final design.

D. State Metric Initialization

Forcing a known starting state by appropriate initialization of state metrics is useful for block decoding and selftest. In order for the dynamic range bound given in (4) to hold, the initial state metrics must satisfy the constraints imposed by the trellis structure and the update algorithm. For example, state-0 and state-1 have a common ancestor state one iteration back that constrains the state metrics to differ at most by the maximum branch metric.

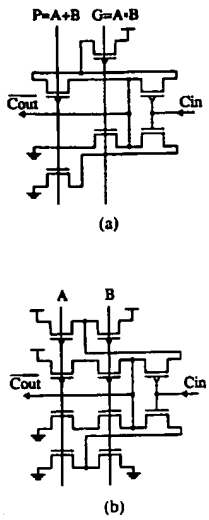

Fig. 9. (a) Adder carry chain circuit. (b) Subtractor carry chain circuit.

Valid starting metrics were found by simulation of the decoder with all zero inputs until steady-state metrics were reached. At this point state-0 is the current and most likely state and all other metrics have reached a maximum value under the constraints imposed by the trellis. The initial state metrics for the 32-state decoder are given in Table II.

TABLE II
32-STATE ACS INITIALIZATION

State	0	1	2	3	4	5	6	7
Init. Metric	0	14	28	14	35	35	21	21
State	8	9	10	11	12	13	14	15
Init. Metric	35	35	35	35	28	28	28	28
State	16	17	18	19	20	21	22	23
Init. Metric	42	42	42	42	35	35	35	35
State	24	25	26	27	28	29	30	31
Init. Metric	35	35	35	35	28	28	28	28

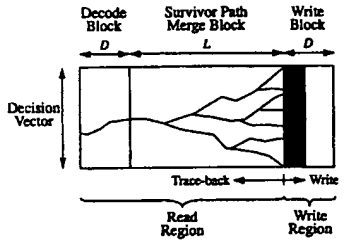

Fig. 10. Decision memory organization.

E. Trace-back Unit

In order to match the throughput of the trace-back unit to that of the ACS array, implementation of the trace-back algorithm [15] described in Section II-A requires L/k trace-back recursions per ACS iteration, where L is the survivor path length and decisions are based on a radix-2^k trellis. For the 32-state radix-4 decoder implemented, $L = 32$, $k = 2$, and the required trace-back recursion rate (TRR) is 16 per ACS iteration. For typical implementations the recursion rate is limited to less than two.

The block-based architecture in [16] achieves a TRR of one using four independent single-ported decision memories, each of depth L. This architecture has been generalized into a class of multiple read-pointer architectures (equivalent to multiple trace-back units) [17]. Using six independent memories, the total memory depth can be reduced to $3L$ ($L/2$ per memory). By increasing the radix of the trellis using pretraced decisions as described in Section II-C, we are able to implement the simplest block-based architecture using one single-ported decision memory of size $3L$ operating at a TRR of one.

The decision memory is organized as a cyclic buffer and is conceptually partitioned into a read and write region as shown in Fig. 10. During each block decode phase, new pretraced decisions are written to the write region on even clock cycles, while the survivor path is traced and decoded from the read region on odd clock cycles (a clock is equivalent to a single radix-4 ACS iteration). Trace-back of the read region proceeds to a depth L, the final state of which is used to initialize the trace-back and decode of the remaining block of length D. Once a block is decoded its decisions can be discarded and it becomes the write region of the next phase.

Given the constraint that the block write and trace-back operations must complete in the same time interval, it can be easily shown that the required TRR is given by

$$\mathrm{TRR} = 1/2\left(1 + \frac{L}{D}\right). \tag{6}$$

For each set of read vectors, the trace-back recursions have two cycles to complete, thus restricting the TRR to be a multiple of $1/2$. Values of D that are of practical interest are L and $L/2$, corresponding to a total decision memory depths of $3L$ and $2L$, respectively. Based on area estimates for the decision memory and periphery logic, the $3L$ architecture was 20% smaller than the $2L$. This apparent paradox is due to the fact that the additional peripheral logic required to support 1.5 trace-back recursions per cycle more than offset the area gains achieved by reducing the decision memory depth. Given the smaller area and slower TRR of one, the $3L$ architecture was chosen for implementation.

The block diagram of the complete decision memory and trace-back unit is shown in Fig. 11. Pretraced decisions from the ACS array are stored as 32-state by 4-b vectors in the decision memory prior to trace-back. Given a survivor path length of 32 for the radix-2 trellis, the effective radix-16 survivor path length is equal to eight vectors, and the total memory size is 24 vectors. Each decision memory word contains two vectors, which allows a double vector read. Decision vector writes alter-

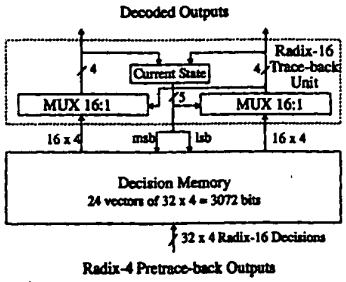

Fig. 11. Trace-back unit.

nate between the left and right halves of the memory. Read cycles fetch two successive decision vectors in parallel, which are then processed in the trace-back unit.

Each trace-back recursion consists of a 32-way selection from the read decision vector based on the current 5-b state. The selected 4-b decision and the current state are combined according to (3) to form the next state. The timing of the trace-back critical path is such that the updated state is available before the end of the next read cycle. This is exploited by completing one level of the 32-way selection during the read. The MSB and LSB of the current state are used as column address bits, selecting even or odd states from the left and right halves of the decision memory. This reduces the trace-back recursion selection to 1 of 16 states using a 16:1 multiplexer.

By combining the cyclic buffer memory management scheme and pretraced decisions, we achieved a trace-back throughput matched to the ACS array in an area-efficient manner using a compact single-ported decision memory.

F. Decoder Output LIFO

The block based trace-back algorithm naturally produces a decode sequence which is time reversed within each block. Correct temporal ordering of the decoder output is achieved using two last-in first-out (LIFO) buffers that double buffer each decoded block.

G. Self-Test Unit

The convolutional encoder was also implemented on the same chip for completeness and to facilitate an extensive on-chip self-test. In self-test mode, the encoder input is switched to a pseudorandom sequence generator. The binary encoder output symbols are converted to a 3-b soft decisions for input to the decoder. The mapping is chosen to ensure error-free decoding; hence, a delayed version of the original encoder input can be compared to the decoder output to verify functionality. The soft-decision mapping chosen was $0 \rightarrow 011$ and $1 \rightarrow 100$. This noisy mapping was chosen over the obvious noiseless bit extension because it ensures nonzero branch metrics and state metric growth.

The length of the pseudorandom sequence generator was chosen to ensure extensive test coverage of the decision memory. When operating correctly in self-test mode, it can be asserted that each decoded 4-b decision is both written and read correctly. The 20-tap pseudorandom sequence generator ensures that decoded state sequence addresses every 4-b decision location with all possible bit patterns.

H. Clocking and I/O

A single-phase clocking strategy was chosen to simplify clock distribution while maximizing system throughput. All latch stages are implemented using rising edge true-single-phase circuits [18]. The global clock is distributed up the center of the chip, from which local module clocks are derived using two-stage buffers of identical delay.

To avoid running the external I/O at the internal clock rate of 70 MHz, the input stream is parallelized to run at half the internal clock rate. For consistency a half-rate clock is supplied to the chip and internally doubled in rate. At the external clock rate of 35 MHz, four input symbol pairs $G_1 G_2$ are input per cycle and four decoded bits are output per cycle, corresponding to a throughput of 140 Mb/s.

IV. Radix-4 ACS Placement and Routing

In fully parallel decoders the state metric interconnect represents a significant portion of the ACS array area. The radix-4 trellis interconnect, with four inputs per ACS, appears as though it might be twice as complex as a radix-2 trellis. However, by arranging the four-way ACS units into radix-4 units as shown in Fig. 4, and considering the ACS array as an interconnect of such radix-4 units, the number of global state metric transfers is equal to the number of states, which is the same as that achieved in radix-2 arrays.

An area-sufficient topology for radix-2 arrays is a ring of radix-2 units arranged in two columns [19]. Each node has an output that connects directly to an adjacent unit, while the other output connects to a central global state metric routing channel. Extending this topology to the radix-4 case results in 75% of the global state metric connections being resolved in the central route compared to 50% for the radix-2 case. The following describes a modification of the ring topology for a radix-4 array, which is both area efficient and regular in routing structure. In fact, the regularity was such that the ACS array layout was produced entirely using a tile based generator, without the need for a channel router.

A. Radix-4 ACS Ring Topology

The global state metric interconnect can be described by a directed graph. Vertices correspond to radix-4 ACS units and edges correspond to the global state metric transfers. An example of the interconnect graph for a 16-state radix-4 trellis is shown in Fig. 12(a).

Radix-4 units are organized into a ring, such that the sequence of units around the ring corresponds to a Hamilton cycle in the interconnect graph. This guarantees that

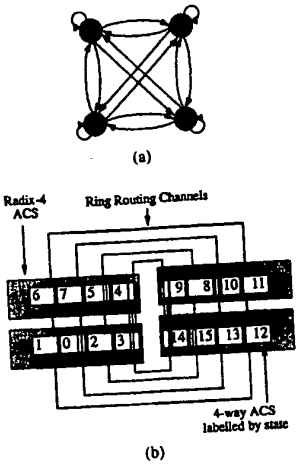

Fig. 12. (a) 16-state radix-4 ACS connect graph. (b) Ring topology and placement.

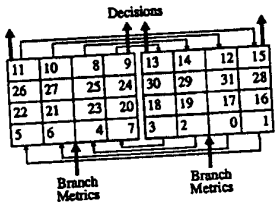

Fig. 13. 32-state ACS placement.

each unit has an output that connects to the adjacent unit. In the 16-state example, the sequence (0, 1, 2, 3) is an obvious cycle and the ring is physically organized as two columns of radix-4 units as shown in Fig. 12(b). In this case the radix-4 units have been expanded to show the four constituent ACS units and the associated state labels.

The modification of the radix-2 topology is to assign four ring routing channels within the array, one ring per ACS column as shown in Fig. 12(b). ACS units with outputs that simply abut to the adjacent unit are assigned to the outer column to minimize the effect of long end routes. The physical position of the remaining three ACS units is unconstrained and hence can be optimized to minimize the total routing channel width and to match longer routes with shorter end routes. The final minimum channel width topology is chosen from a search space consisting of all Hamilton cycles in the radix-4 interconnect graph.

B. 32-State ACS Topology

The ACS placement for the 32-state decoder implemented is shown in Fig. 13. The total width of ring routing channels is ten state metric buses. This solution was chosen over solutions requiring nine buses due to the high degree of regularity in the resulting routes, which made tile-based generation of layout easier. Branch metrics are routed vertically up the middle of the left and right columns, from which the required metrics are tapped off horizontally to ACS units within each radix-4 unit. The radix-4 ACS ring topology and routing strategy is clearly visible in the complete chip micrograph shown in Fig. 14. Partitioning of the 32-state ACS array by area is 53% ACS logic and 33% state metric routing, which is comparable to 57% ACS logic and 32% state metric routing achieved in the area-efficient radix-2 array [19].

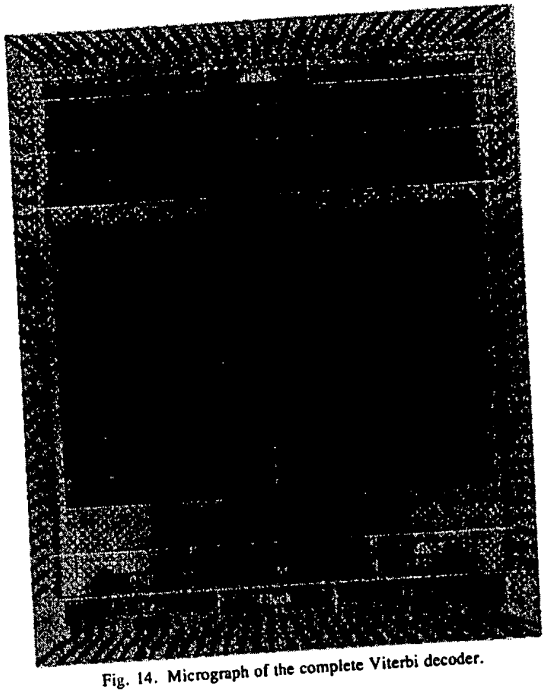

Fig. 14. Micrograph of the complete Viterbi decoder.

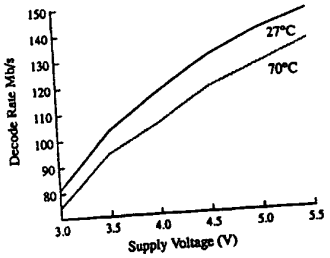

Fig. 15. Measured performance versus supply voltage.

V. Fabrication and Test Results

The design was fabricated using 1.2-μm double-metal CMOS at Hewlett-Packard via MOSIS on the MAY-1991 run. A micrograph of the complete chip is shown in Fig. 14. The 7.30-mm × 8.49-mm chip contains 146 000 transistors and is packaged in a 132-pin PGA.

The measured throughput as a function of supply voltage is summarized in Fig. 15. Typical parts are fully functional at an internal clock rate of 70 MHz and dissi-

pate 1.8 W at $V_{DD} = 5.0$ V and $T_A = 27°C$. This corresponds to a radix-4 iteration rate of 70 MHz, equivalent to a radix-2 decode rate of 140 Mb/s.

VI. Summary

We have shown that the throughput of a radix-2 state-parallel Viterbi decoder can be increased using higher radix formulations of the trellis. The radix-4 architecture is the radix of choice because it is the only architecture to offer ideal linear scaling and a practically achievable throughput increase. Throughput doubling relies on implementing a 4-way ACS operation at the same rate as a 2-way ACS. A 4-way ACS circuit has been presented that achieves this goal to within 17%, resulting in an overall speedup of a factor of 1.7.

A preprocessing stage has been added to the trace-back portion of the Viterbi algorithm, which further collapses the radix-4 trellis into a radix-16 structure. Using pre-traced decisions, implementation of the trace-back algorithm is possible without fragmenting the decision memory into multiple memories and without running multiple trace-back recursions in parallel. The radix-16 trace-back architecture is implemented using a single-ported decision memory of depth equal to three times the survivor path length.

The state metric and branch metric routing of a radix-4 ACS array represents a significant proportion of the array area. An area-efficient ring-based radix-4 ACS placement and routing topology has been proposed, resulting in a routing overhead comparable to radix-2 designs.

These concepts are demonstrated in a 32-state $R = 1/2$ Viterbi decoder, implemented using 1.2-μm CMOS technology. A radix-4 iteration rate of 70 MHz corresponding to an effective radix-2 decode rate of 140 Mb/s was achieved. This represents a significant increase over current commercial radix-2 designs of equivalent complexity that have been limited to a decode rate of 25 Mb/s.

References

[1] "20 Mbps convolutional encoder Viterbi decoder STEL-2020," Stanford Telecommunications, Santa Clara, CA, Oct. 1989.

[2] "The Q1650-3L Viterbi codec," Qualcomm Inc., San Diego, CA, 1991.

[3] H. F. Lin and D. G. Messerschmitt, "Algorithms and architectures for concurrent Viterbi decoding," in *Proc. ICC*, 1989, pp. 836-840.

[4] G. Fettweis and H. Meyr, "A modular variable rate Viterbi decoding implementation for high data rates," in *Proc. EUSIPCO*, vol. 1, Sept. 1988, pp. 339-342.

[5] P. J. Black and T. H.-Y. Meng, "A hardware efficient parallel Viterbi algorithm," in *Proc. ICASSP*, vol. 2, Apr. 1990, pp. 893-896.

[6] G. Fettweis and H. Meyr, "High-speed parallel Viterbi decoding: Algorithm and VLSI architecture," *IEEE Commun. Mag.*, vol. 29, no. 5, pp. 46-55, May 1991.

[7] P. J. Black and T. H.-Y. Meng, "A 140Mb/s, 32-state, radix-4 Viterbi decoder," in *ISSCC Dig. Tech. Paper*, Feb. 1992, pp. 70-71.

[8] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268-278, Mar. 1973.

[9] P. J. Black and T. H.-Y. Meng, "A unified approach to the Viterbi algorithm state metric update for shift register processes," in *Proc. ICASSP*, vol. 5, Mar. 1992, pp. 629-632.

[10] P. G. Gulak and T. Kailath, "Locally connected VLSI architectures for the Viterbi algorithm," *IEEE J. Selected Areas Commun.*, vol. 6, no. 3, pp. 526-537, Apr. 1988.

[11] H. Thapar and J. Cioffi, "A block processing method for designing high-speed Viterbi detectors," in *Proc. ICC*, vol. 2, June 1989, pp. 1096-1100.

[12] A. P. Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol. 37, no. 11, pp. 1220-1222, Nov. 1989.

[13] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*. New York: McGraw-Hill, 1979, pp. 258-261.

[14] J. A. Heller and I. M. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. Commun. Technol.*, vol. COM-19, no. 5, pp. 835-848, Oct. 1971.

[15] C. M. Radar, "Memory management in a Viterbi decoder," *IEEE Trans. Commun.*, vol. COM-29, no. 9, pp. 1399-1401, Sept. 1981.

[16] H. A. Bustamente, I. Kang, C. Nguyen, and R. E. Peile, "Stanford telecom VLSI design of a convolutional decoder," in *Proc. MILCOM*, vol. 1, pp. 171-178, Oct. 1989.

[17] R. Cypher and C. B. Shung, "Generalized trace-back techniques for survivor memory management in the Viterbi algorithm," in *Proc. GLOBECOM*, vol. 2, pp. 1318-1322, Dec. 1990.

[18] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE J. Solid-State Circuits*, vol. 24, no. 1, pp. 62-70, Feb. 1989.

[19] J. Sparso, H. N. Jorgensen, E. Paaske, S. Pedersen, and T. Rubner-Petersen, "An area-efficient topology for VSLI implementation of Viterbi decoders and other shuffle-exchange type structures," *IEEE J. Solid-State Circuits*, vol. 26, no. 2, pp. 90-97, Feb. 1991.

Peter J. Black (S'88) was born in Brisbane, Australia, in 1964. He received the B.E. degree in electronic engineering from the University of Queensland, Australia, in 1985. In 1990 he received the M.S.E.E. degree from Stanford University, Stanford, CA, and is currently a Ph.D. candidate at the same university.

From 1986 to 1988 he worked at Austek Microsystems, Adelaide, Australia, on custom VLSI design for digital signal processing.

Mr. Black is a Fulbright Scholar and received an IBM Graduate Fellowship in 1990. In 1985 he was awarded the University Medal by the University of Queensland.

Teresa H. Meng received the B.S. degree from National Taiwan University in 1983 and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1984 and 1988, respectively.

She has been an Assistant Professor in the Department of Electrical Engineering at Stanford University, Stanford, CA, since 1988. Her current research activities include digital signal processing, wireless data communication, and dedicated-hardware design of real-time DSP applications.

Dr. Meng received the IEEE Signal Processing Society's Paper Award in 1987, the 1989 NSF Presidential Young Investigator Award, the 1989 ONR Young Investigator Award, a 1989 IBM Faculty Development Award, and the 1988 Eli Jury Award at the University of California, Berkeley, for recognition of excellence in systems research.